Ken Amamori
4/27/2021

CS 4375
Professor Ward

File I/O Assignment

As a way to send secret messages, I've invented a way to hide the payload in a large file, padded by thousands or millions of junk characters.

To extract the secret message, look for the eight-character string *uteputep* and then extract information using the words that follow it. Specifically

1. The 3 characters after the first *uteputep* are a count of the total number of *uteputep* occurrences in the file.
2. The 3 characters after the second *uteputep* represent the length of the payload.
3. The payload itself occurs after the second-to-last occurrence of *uteputep*.

For example, to convey the secret message *miners*, I could send you the file:

garbage uteputep004 garbageuteputep006 gg uteputepminers garbage uteputepgg

In general, the files will be much larger, to make it harder for eavesdroppers to infer the message. The occurrences of *uteputep* will be roughly evenly distributed throughout the files.

Baseline code for extracting these messages is in the file utepDecode.py, but this is slow.

Part 1

Design a version that uses python's seek() to find the messages faster. Do not plan to simply increase the buffer size: use no buffers larger than 512 bytes.

a. Write a short description of your planned algorithm. This may be done in pseudocode, by specifying your planned changes to the baseline code, with a diagram, etc. This should not exceed the equivalent of two or three sentences.

```
buffer = 512
key = b'uteputep'

f1 = open file
f2 = open file
i1 = 0
i2 = 0
f1_o = 0
f2_o = 0

do-while i1 is less than i2 == (i1<i2)
  if data1 = key
    f1_o ++
  if f1_o is -1
```

```
        pass
    elif f1_o is not 2
        data1 = f1.seek(-len(key), 1)
    else
        data1 = f1.seek(-3, 1)
        f1_o = -1

    if data2 = key
        f2_o ++
    if f2_o is -1
        pass
    elif f2_o is not 2
        data2 = f2.seek(len(key), 1)
    else
        if f1_o is 2 or f1_o is -1
            data2 = f2.seek(-data1, 1)
            return data2
do
    data1 = f1.seek(-len(key))
    data2 = f2.seek(len(key), 2)
```

b. Come to class prepared to present your approach to the class or a subgroup.

Part 2

a. Implement a faster version. Show that your version works on the test cases.

b. Generate a larger test file with `utepEncode.py lane8 100 2000000`. The file should be about 200 MB. Show that your new version works for this file.

c. Time the original utepDecode.py and your version on this large file. Report both times and the percentage speedup obtained.

d. Describe the machine you ran this on, including everything you think would be relevant to the speeds.

e. Acknowledge anyone or anything that was helpful to you. This may include classmates who discussed the problem with you or code that you found. (Note that this problem may resemble that of reimplementing the unix "tail" command.)

Note: Those whose code is fastest, clearest, and best described may be invited to present it to the class for a free pass on any past or future assignment.